

In the Claims:

1. (Canceled) A method for developing an application program, the application program using at least one resource file, the resource file including resource data in a markup language to be interpreted in accordance with a schema, wherein the resource file does not explicitly specify all resource data, the method comprising the steps of: identifying a first resource element data needed for the first resource element; identifying specified data for the first resource element in the resource file; and obtaining non-specified resource data, which is not explicitly specified for the first resource element in the resource file, from one of the sources in the group consisting of resource data for an implementation of a second resource element, resource data from a default style specification in the schema, resource data from a parent element of the first resource element, resource data specified by the application program, and resource data specified in programmable settings.

2. (Canceled) The method of claim 1 further having the step of using a software tool for editing a first resource file corresponding to the application being developed, wherein the software tool uses a markup language to write new information in the first resource file.

3. (Canceled) The method of claim 1 further having the step of naming the resource file in accordance with a naming scheme to aid in identifying the resource data in the resource file.

4. (Canceled) The method of claim 1 wherein furthermore the resource file is external to the application program.

5. (Canceled) The method of claim 1 wherein the step of obtaining data includes getting default resource data from programmable settings chosen by a user in setting up a computing environment for executing the application program.

6. (Canceled) The method of claim 1 wherein the step of obtaining data includes getting default resource data from resource data used to implement the parent node of a current node in a tree formed by parsing the markup language in the resource file wherein the current node corresponds to the first resource element.

7. (Canceled) The method of claim 1 wherein the step of obtaining data includes getting default resource data from the default style specification for a graphical user interface.

8. (Canceled) The method of claim 1 wherein the step of obtaining data includes getting default resource data from data used to implement the parent node of a current node in a tree formed by parsing the markup language in the resource file wherein the current node corresponds to the first resource element.

9. (Canceled) A system for developing an application program in an environment including developers for modifying code, designers for evaluating aesthetic and ease of use of a user interface for the application program, the system comprising: a resource file containing information encoded in accordance with a markup language; a resource-loader routine in the operating system for retrieving information from the resource file responsively to a resource-requesting call made by the application program and obtaining a tree corresponding to the markup language; an interpreter for rendering parsed markup language encoded information in the resource file; and at least one default source of resource information wherein the resource information is not explicitly specified in the resource file.

10. (Canceled) The system of claim 9 wherein a schema for the markup language is a first default source of resource information.

11. (Canceled) The system of claim 9 wherein a style specification is a first default source of resource information.

12. (Canceled) The system of claim 9 wherein an implementation of a resource element is a first default source of resource information.

13. (Canceled) The system of claim 12 wherein the first default information includes reference values for implementing explicitly specified resource information in the resource file.

14. (Canceled) The system of claim 9 wherein a parent node of a current node in a tree corresponding to the markup language in the resource file is a first default source of resource information.

15. (Canceled) The system of claim 9 wherein at least one programmable setting is a first default source of resource information.

16. (Canceled) The system of claim 9 wherein the application program is a first default source of resource information not found in the resource file.

17. (Canceled) The system of claim 9 wherein the resource loader obtains resource information from a first default source if the resource file includes explicit instructions to use default resource information.

18. (Canceled) Computer readable media storing executable instructions for implementing a method for developing an application program, the application program using at least one resource file, the resource file including resource data in a markup language to be interpreted in accordance with a schema, wherein furthermore the resource file does not specify all of the resource information, the method comprising the steps of: identifying a first resource element data needed for the first resource element; identifying specified data for the first resource element in the resource file; and obtaining non-specified resource data, which is not explicitly specified for the first resource element in the resource file, from one of the sources in the group consisting of resource data for an implementation of a second resource element,

resource data from a default style specification in the schema, resource data from a parent element of the first resource element, resource data specified by the application program, and resource data specified in programmable settings.

19. (Canceled) The computer readable media of claim 18 wherein the step of completing the creation of the first resource using default information includes accessing a style specification.

20. (Canceled) The computer readable media of claim 19 wherein the style specification is provided in the markup language.

21. (Canceled) The computer readable media of claim 18 wherein the step of completing the creation of the first resource using default information includes accessing the implementation of a second resource.

22. (Canceled) The computer readable media of claim 18 wherein the step of obtaining data includes getting default resource data from programmable settings chosen by a user in setting up a computing environment for executing the application program.

23. (Canceled) The computer readable media of claim 18 wherein the step of obtaining data includes getting default resource data from the default style specification for a graphical user interface.

24. (Canceled) The computer readable media of claim 18 wherein the step of obtaining data includes getting default resource data from data used to implement the parent node of a current node in a tree formed by parsing the markup language in the resource file wherein the current node corresponds to the first resource element.

25. (Canceled) The computer readable media of claim 18 wherein the step of obtaining data includes getting default resource data from resource data used to implement the parent node

of a current node in a tree formed by parsing the markup language in the resource file wherein the current node corresponds to the first resource element.

26. (New) In an environment of linked computers, a system for collaboratively developing a computer application software product by at least two system users, the system comprising:

a first set of one or more routines used by a first system user for producing an executable program component of the computer application software product;

at least one user interface resource file comprising a document in a markup language, wherein tagged text elements are associated with attributes of a user interface component of the computer application software product; and

a second set of one or more routines used by a second system user for creating and modifying the user interface component by manipulating the at least one user interface resource file, wherein the creating and modifying by the second system user is independent of actions taken by the first system user.

27. (New) The system of claim 26 wherein the routines for creating and modifying the user interface component are used while the computer application software product is being executed, the creating and modifying occurring dynamically, and not requiring a recompilation of the executable program component.

28. (New) The system of claim 26, further comprising a set of operating system resource-loading routines for presenting a user interface corresponding to the user interface component to a third system user.

29. (New) The system of claim 28 wherein the resource-loading routines obtain user interface resource information from a user interface attribute data tree corresponding to the user

interface resource file and, with respect to resource information not specified in the user interface resource file, from a set of default sources of user interface resource information.

30. (New) The system of claim 28 wherein the third system user is also the first system user.

31. (New) The system of claim 28 wherein the third system user is also the second system user.

32. (New) The system of claim 28 wherein the third system user is neither the first system user nor the second system user.

33. (New) A computer-readable medium storing computer-executable instructions and computer-readable data for implementing the system of claim 26.

34. (New) A system for customizing a user interface for an executable computer program by a user of the program, the system comprising:

a set of one or more routines for modifying at least one user interface resource file; and
the at least one user interface resource file, comprising a document in a markup language, wherein tagged text elements are associated with attributes of the user interface.

35. (New) The system of claim 34 wherein the routines for modifying the at least one user interface resource file are invoked while the computer program is being executed, the customizing occurring dynamically.

36. (New) The system of claim 34, further comprising a set of operating system resource-loading routines for presenting the user interface to the user, wherein the resource-loading routines obtain user interface resource information from a user interface attribute data tree corresponding to the user interface resource file and, with respect to resource information

not specified in the user interface resource file, from a set of default sources of user interface resource information.

37. (New) A computer-readable medium storing computer-executable instructions and computer-readable data for implementing the system of claim 34.

38. (New) A method for collaboratively developing a computer application software product by at least two system users, the computer application software product including a user interface, the method comprising:

by a first system user, writing source code for the computer application software product, and generating a first build of the computer application software product; and

by a second system user,

executing the first build, thereby causing the user interface to be presented;

proposing changes to the user interface;

if the proposed changes require the first system user to rewrite the source code and generate a second build, communicating the proposed changes to the first system user; and

if the proposed changes do not require the first system user to rewrite the source code and generate a second build, editing at least one user interface resource file to incorporate the proposed changes, and causing a new user interface to be presented.

39. (New) The method of claim 38 wherein the at least one user interface resource file comprises a document in a markup language, wherein tagged text elements are associated with attributes of the user interface.

40. (New) The method of claim 38 wherein causing the user interface to be presented comprises:

parsing the at least one user interface resource file into a user interface attribute data tree;

invoking operating system resource-loading routines for constructing the user interface;

and

obtaining user interface resource information from the user interface attribute data tree and, with respect to resource information not specified in the user interface resource file, from a set of default sources of user interface resource information.

41. (New) The method of claim 38 wherein causing the user interface to be presented occurs while the first build is being executed and does not require the first build to be re-executed.

42. (New) A computer-readable medium storing computer-executable instructions and computer-readable data for performing the method of claim 38.

43. (New) A method for customizing a user interface for an executable computer program by a user of the program, the method comprising:
executing the computer program, thereby causing the user interface to be presented;
editing at least one user interface resource file, the at least one user interface resource file comprising a document in a markup language, wherein tagged text elements are associated with attributes of the user interface; and
causing a new user interface to be presented.

44. (New) The method of claim 43, wherein causing the user interface to be presented comprises:

parsing the at least one user interface resource file into a user interface attribute data tree;
invoking operating system resource-loading routines for constructing the user interface;
and

obtaining user interface resource information from the user interface attribute data tree
and, with respect to resource information not specified in the user interface resource file, from a
set of default sources of user interface resource information.

45. (New) ~~The method of claim 43 wherein causing the user interface to be presented~~
occurs while the computer program is being executed and does not require the computer program
to be re-executed.

46. (New) ~~A computer-readable medium storing computer-executable instructions~~
and computer-readable data for performing the method of claim 43.